Enabling Practical SDN Security Applications with OFX (The **O**pen**F**low e**X**tension Framework)

John Sonchack, Adam J. Aviv,
Eric Keller, and Jonathan M. Smith

Penn
University of Pennsylvania

University of Colorado
Boulder

USNA
Computer Science and Information Technology

Outline

Introduction

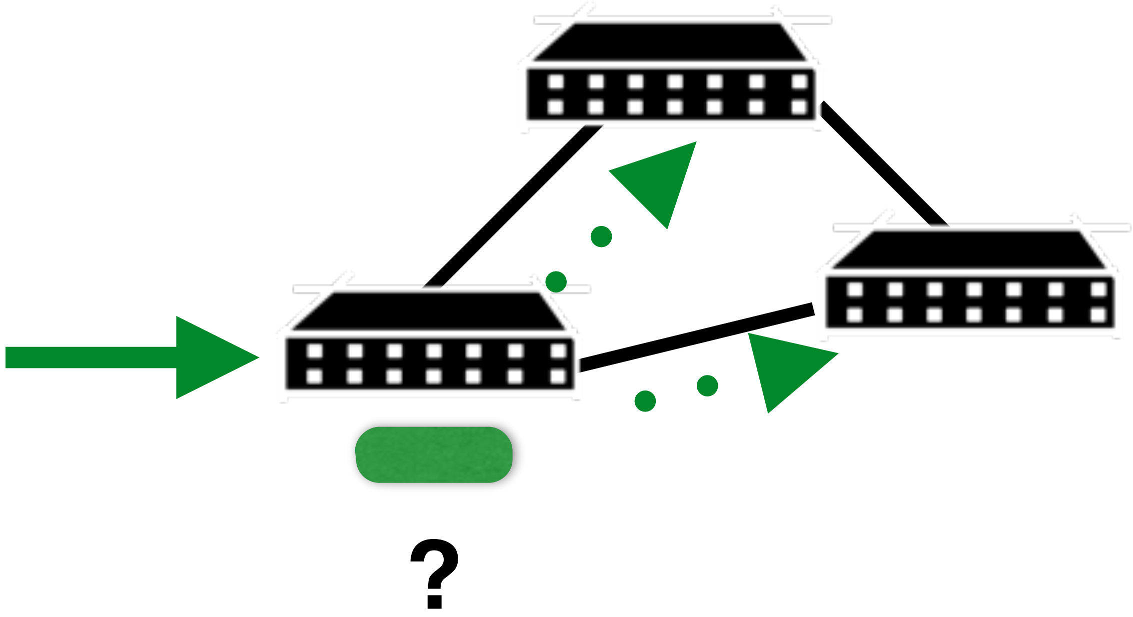
Overview of OFX

Using OFX

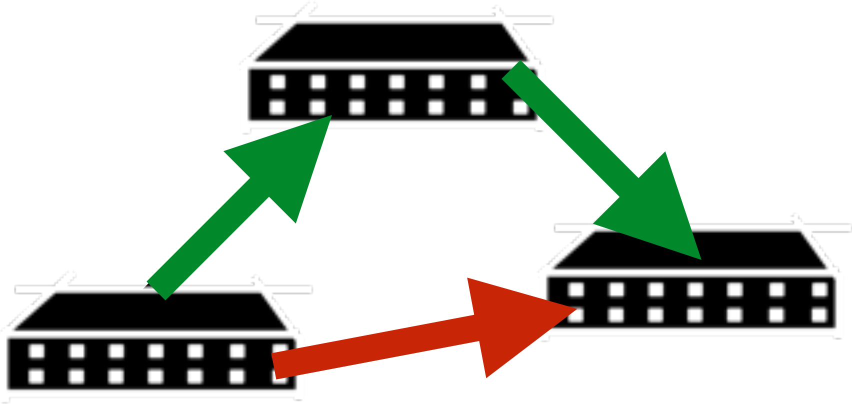
Benchmarks

Basic Networking: Forwarding and Routing

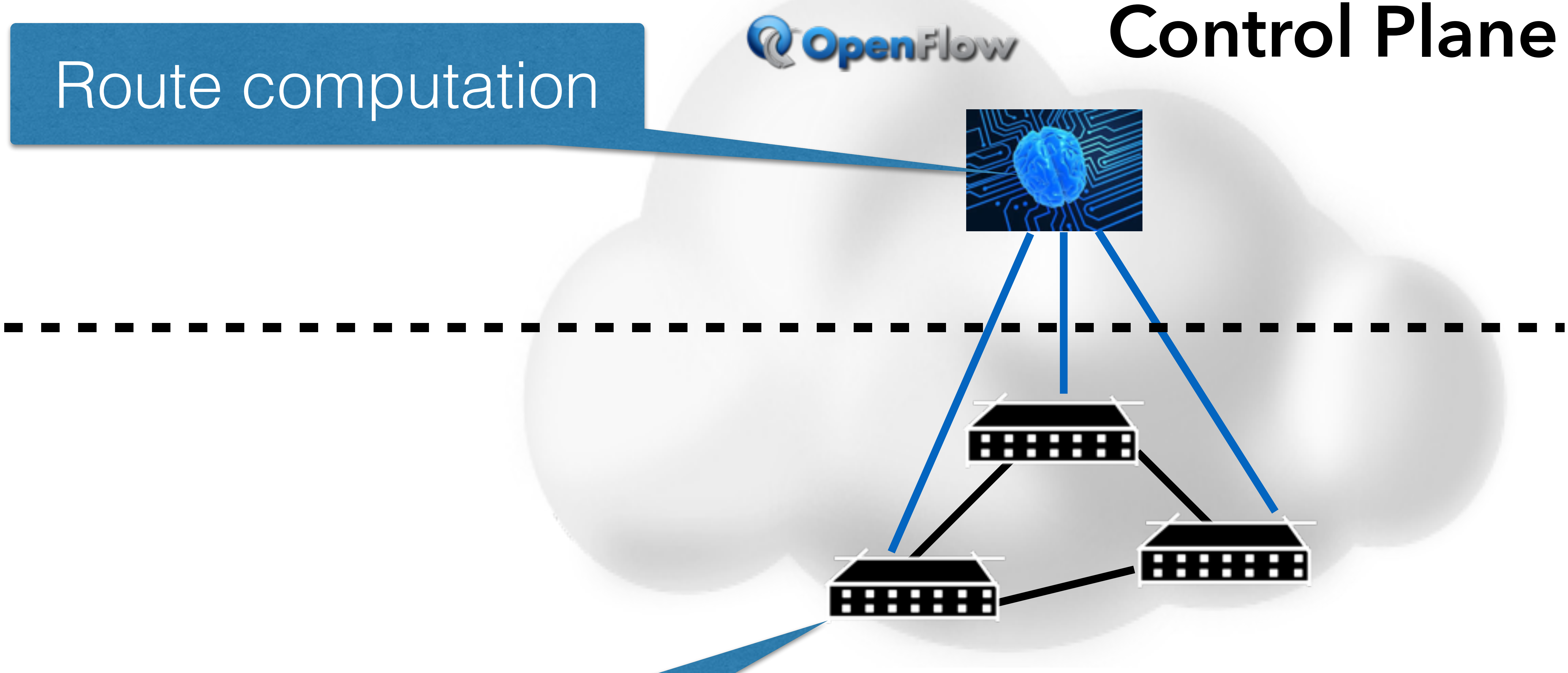
Packet Forwarding



Route Computation



SDNs: Networking in Two Planes



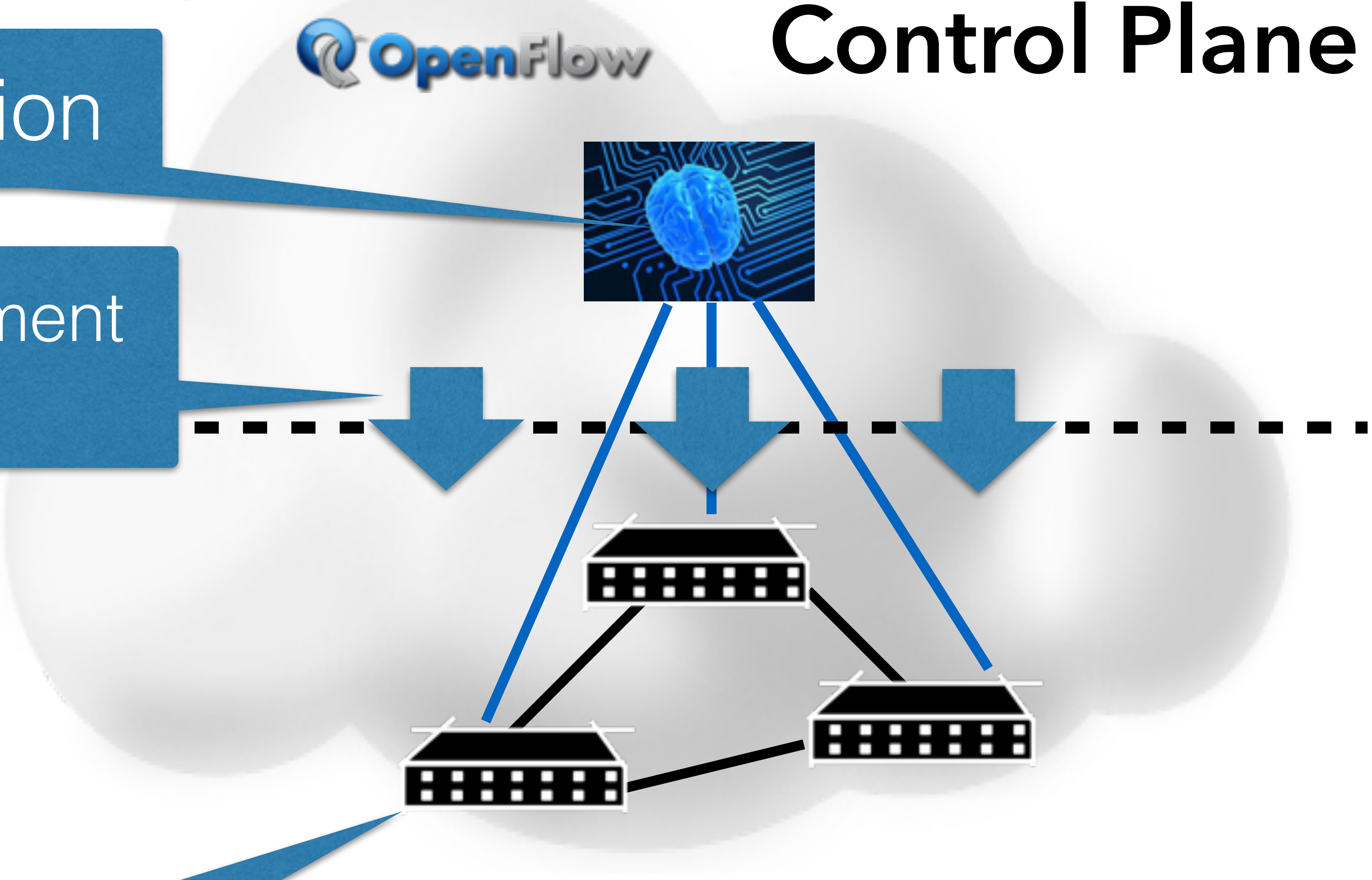
Control Plane

Route computation

Packet forwarding

Data Plane

OpenFlow: A Protocol to Manage Switches



Control Plane

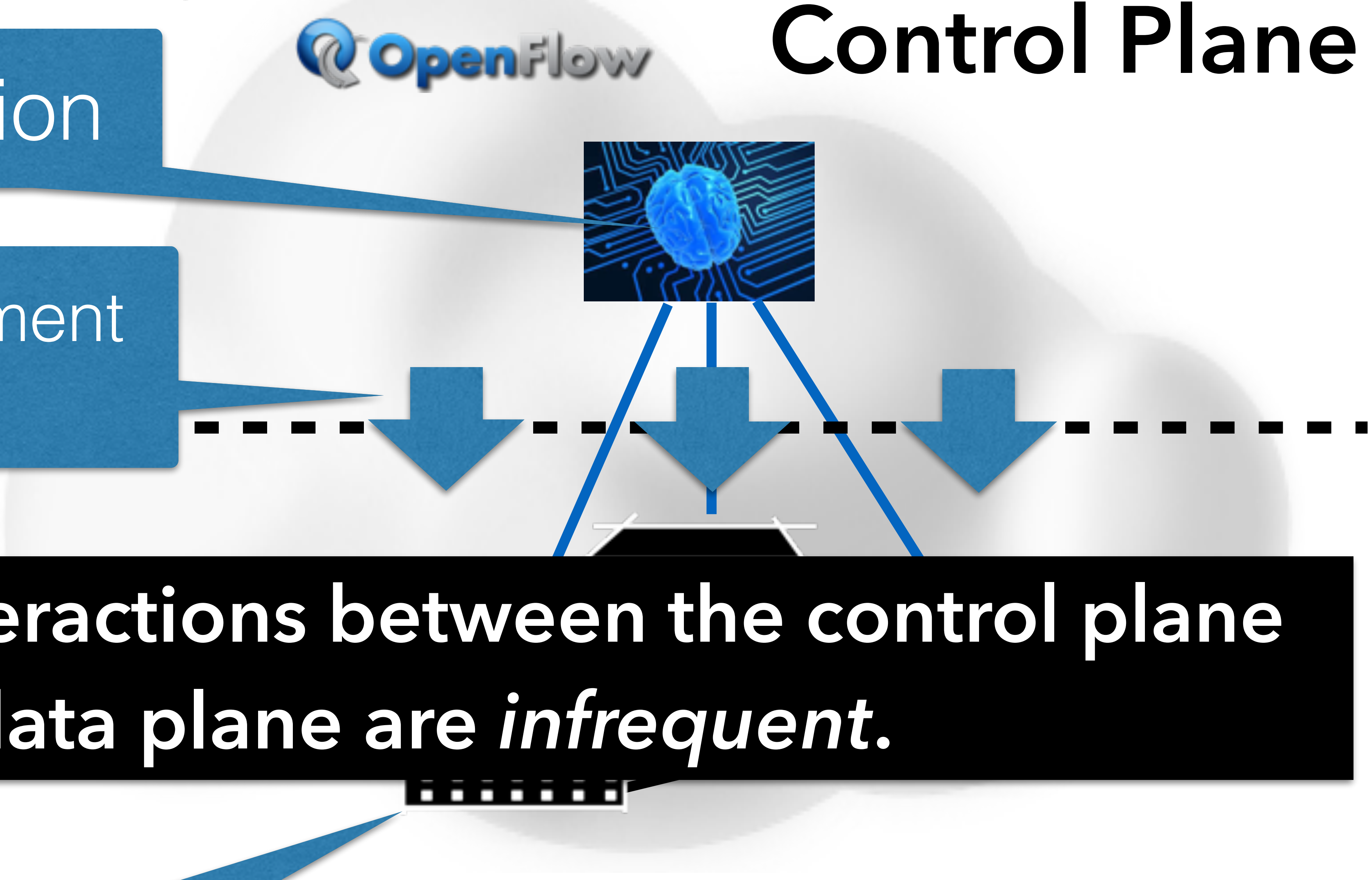
Route computation

Flow rules to implement routes

Packet forwarding

Data Plane

OpenFlow: A Protocol to Manage Switches



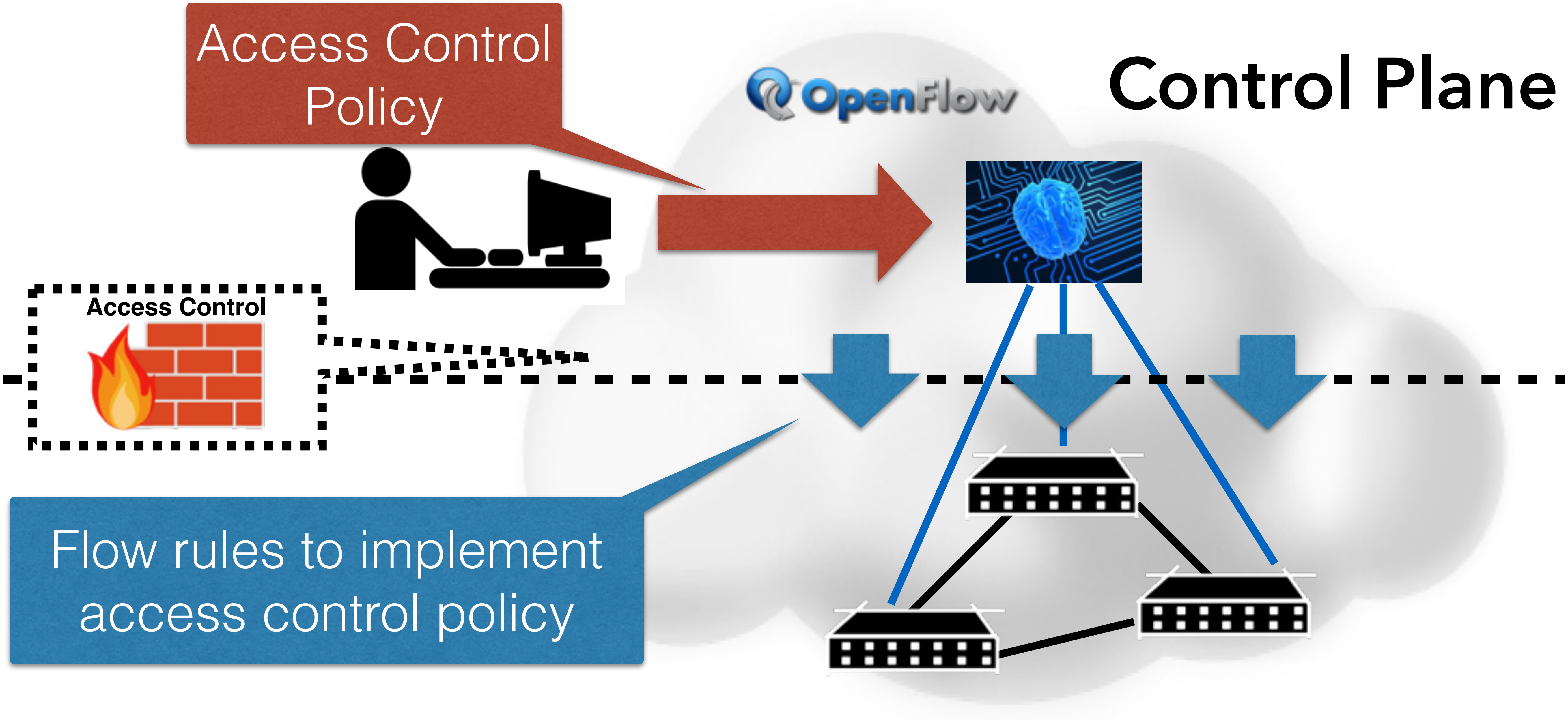
Route computation

Flow rules to implement routes

Assumption: Interactions between the control plane and data plane are *infrequent*.

Packet forwarding

SDNs for Network Security

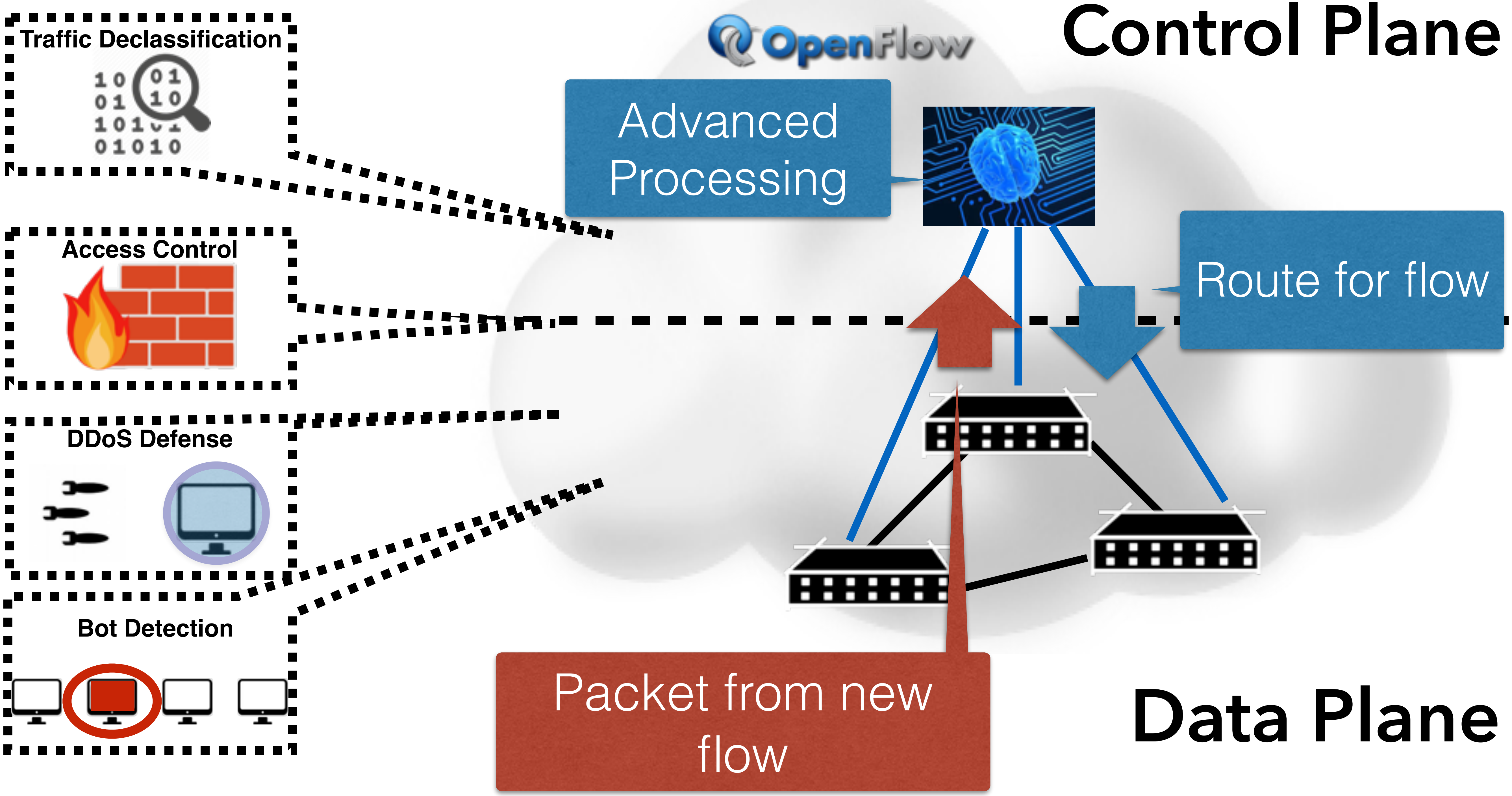


Access Control Policy

Flow rules to implement access control policy

Casado, Martin, et al. **"Ethane: taking control of the enterprise."** *ACM SIGCOMM Computer Communication Review*. Vol. 37. No. 4. ACM, 2007.

SDNs for **Dynamic** Network Security



Traffic Declassification

Access Control

DDoS Defense

Bot Detection

Control Plane

Advanced Processing

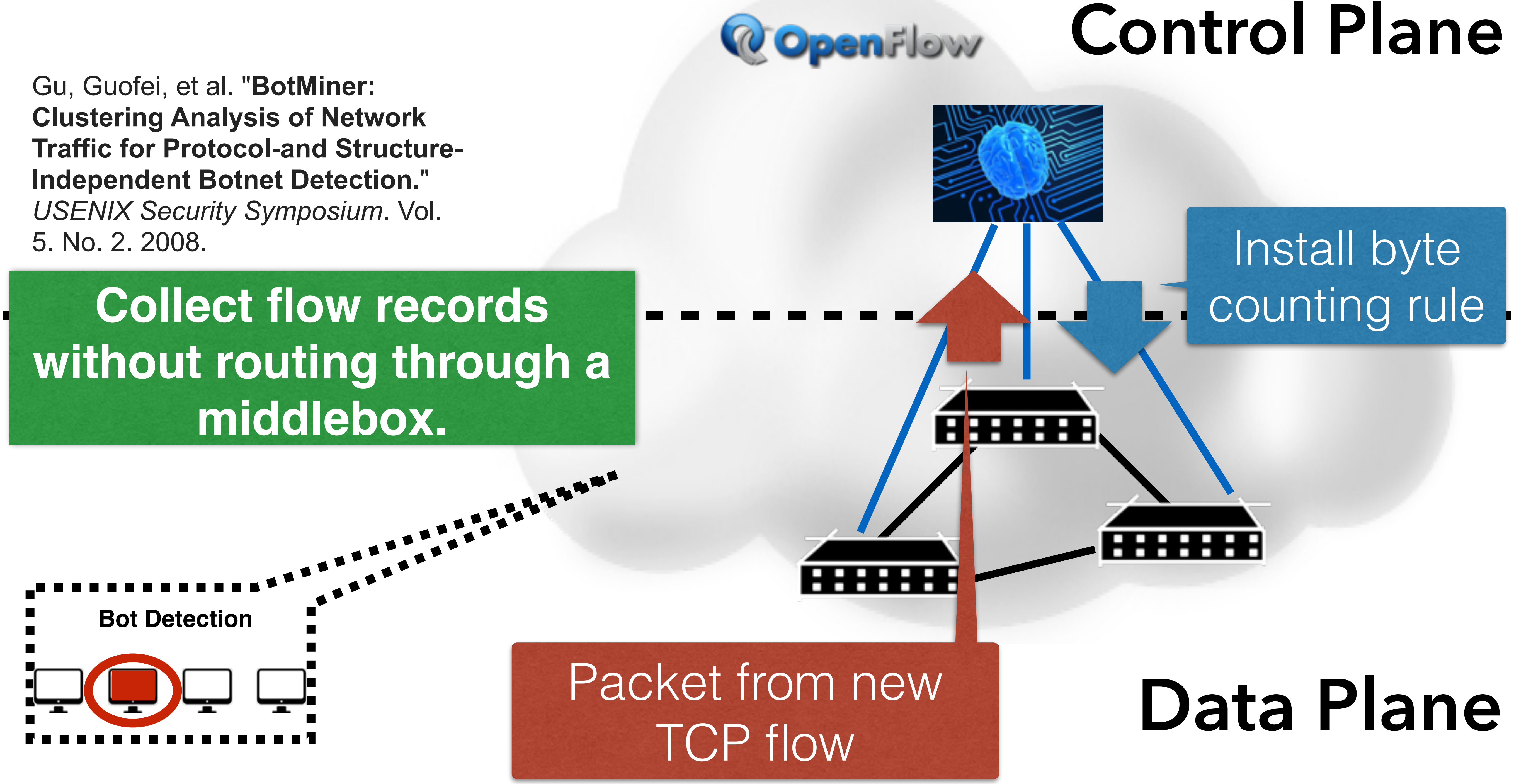
Route for flow

Packet from new flow

Data Plane

SDNs for **Dynamic** Network Security: Flow Monitoring

Gu, Guofei, et al. "**BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection.**" *USENIX Security Symposium*. Vol. 5. No. 2. 2008.

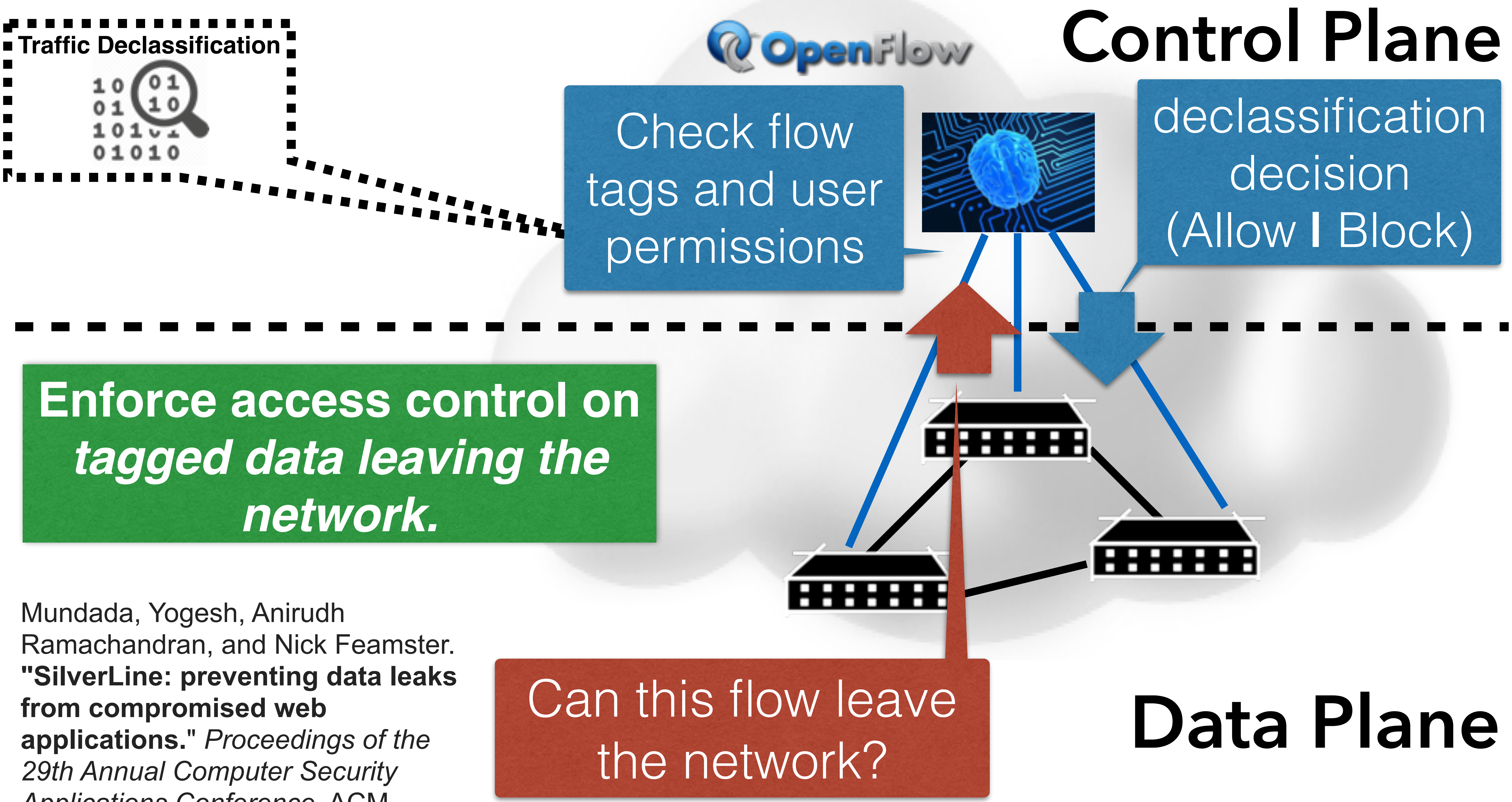


Install byte counting rule

Collect flow records without routing through a middlebox.

Packet from new TCP flow

SDNs for **Dynamic** Network Security: Traffic Declassification



Traffic Declassification

OpenFlow

Control Plane

Check flow tags and user permissions

declassification decision (Allow | Block)

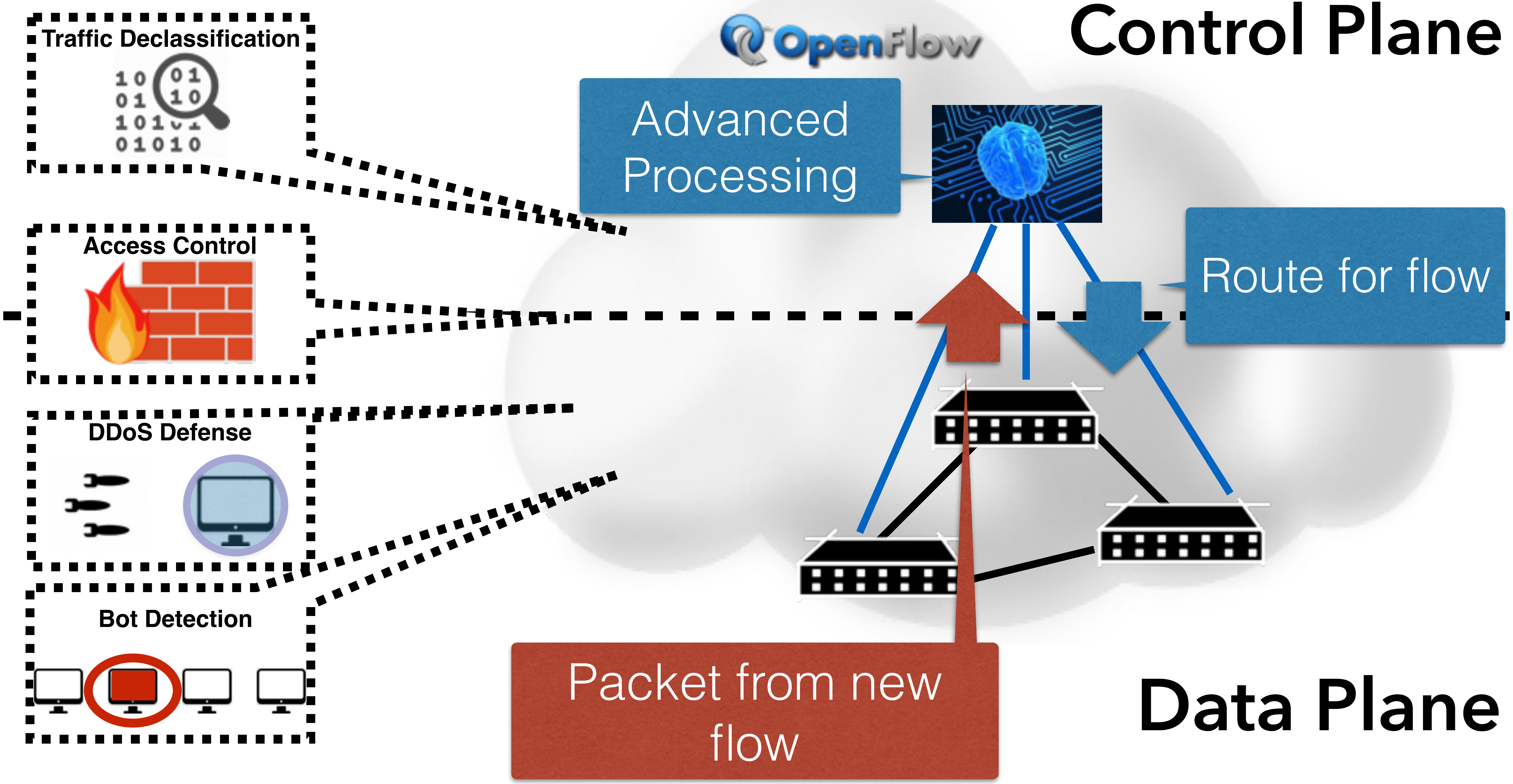
Enforce access control on *tagged data leaving the network.*

Can this flow leave the network?

Data Plane

Mundada, Yogesh, Anirudh Ramachandran, and Nick Feamster. **"SilverLine: preventing data leaks from compromised web applications."** *Proceedings of the 29th Annual Computer Security Applications Conference*. ACM, 2013.

SDNs for **Dynamic** Network Security



Traffic Declassification

Access Control

DDoS Defense

Bot Detection

OpenFlow

Control Plane

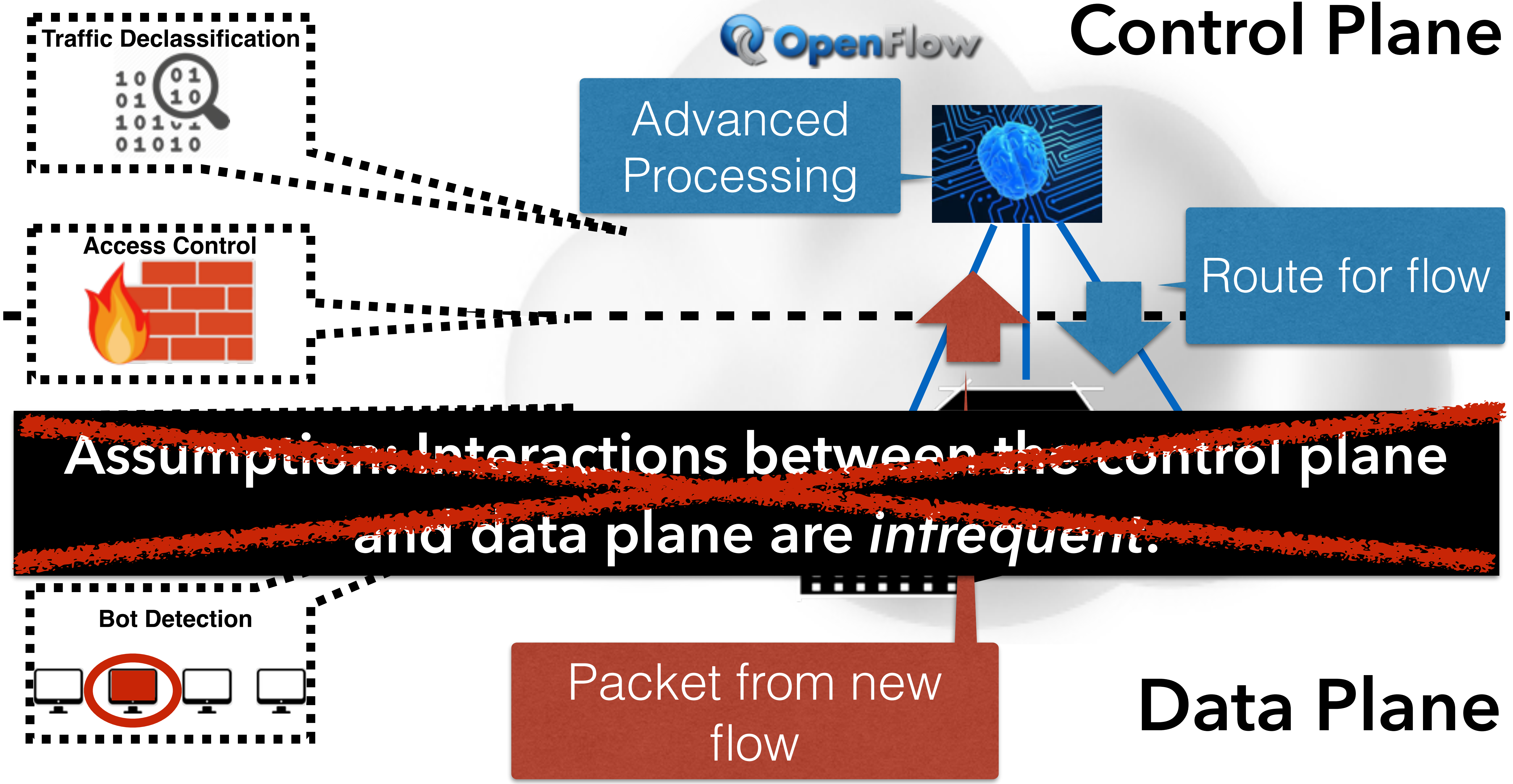
Advanced Processing

Route for flow

Packet from new flow

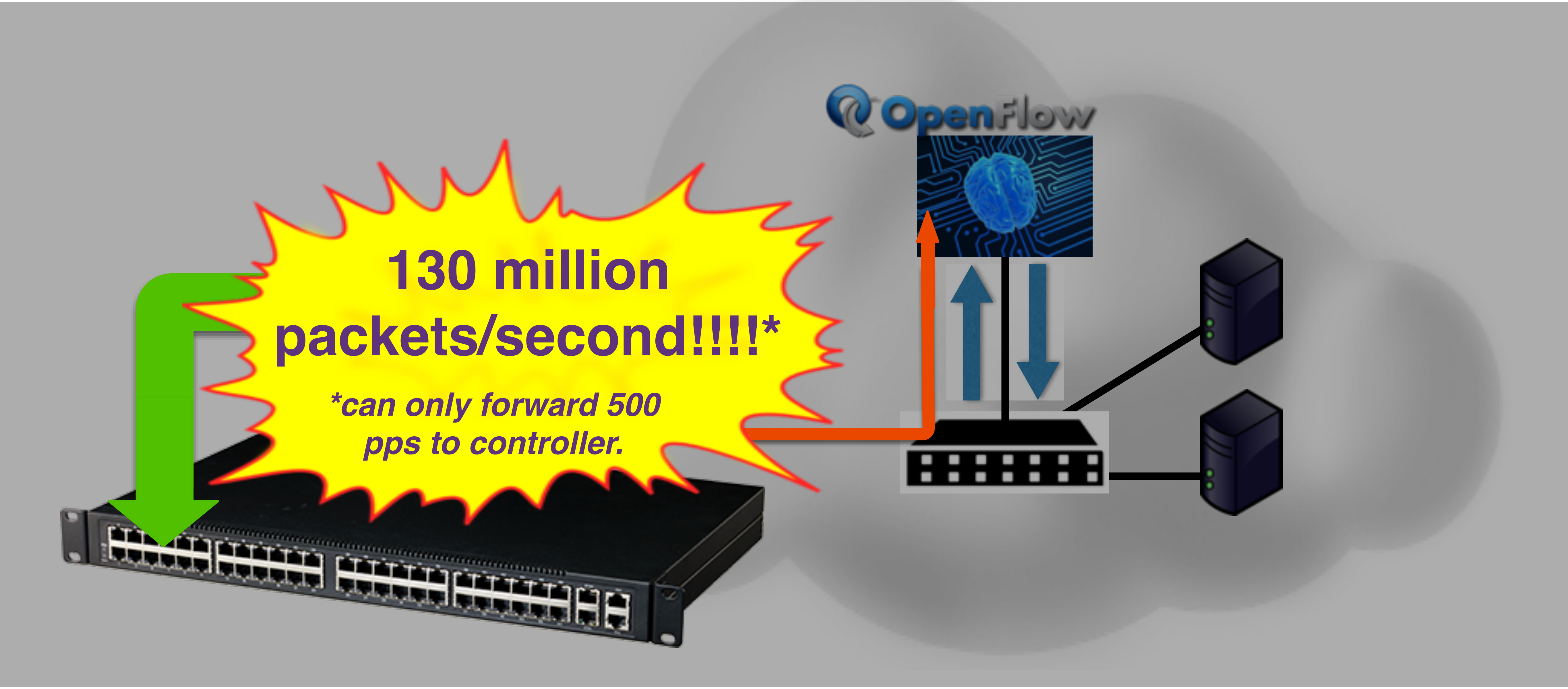
Data Plane

SDNs for **Dynamic** Network Security



~~Assumption: Interactions between the control plane and data plane are *infrequent*.~~

Obstacle: Low Throughput Control Path



130 million packets/second!!!!*

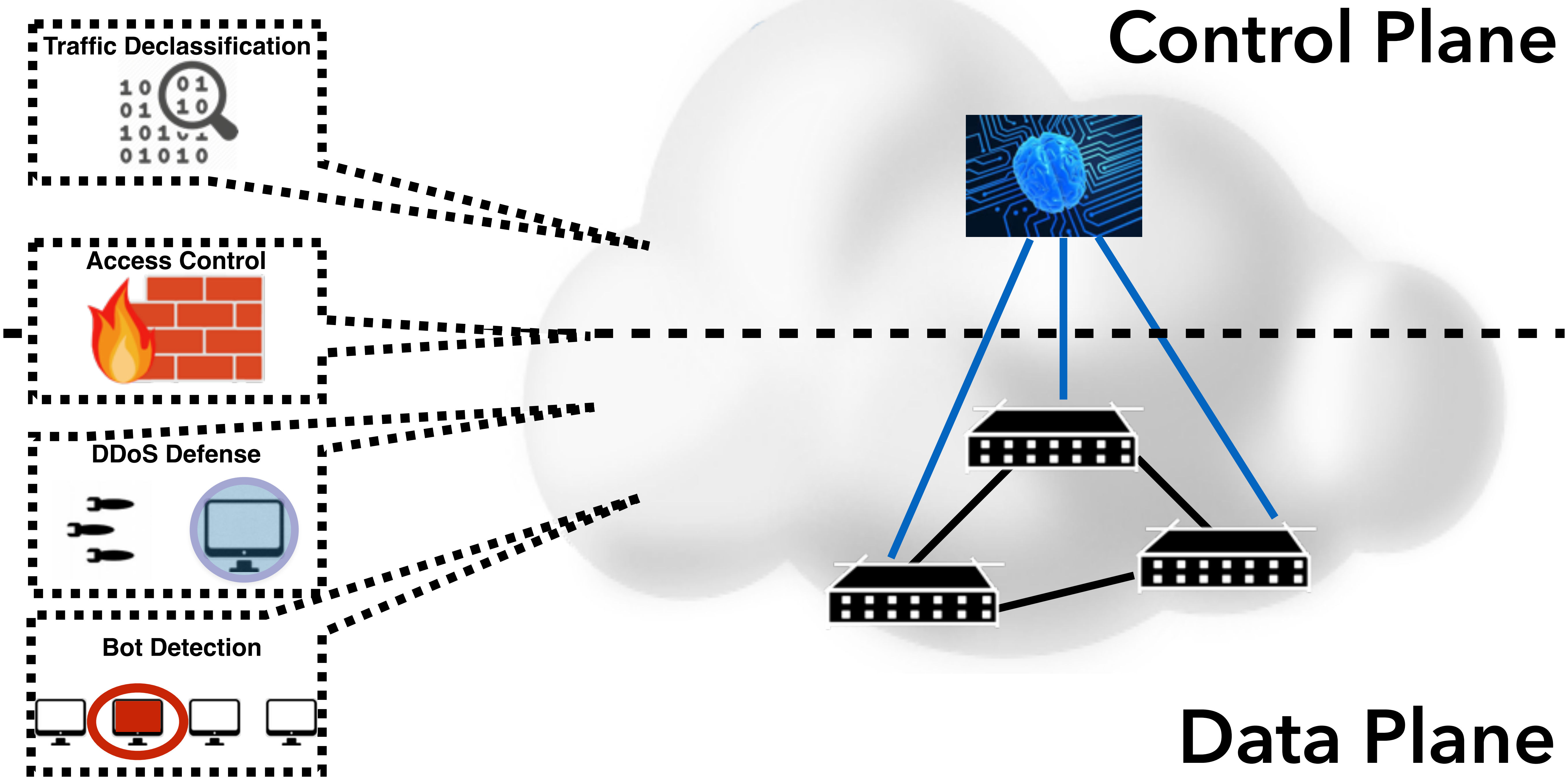
*can only forward 500 pps to controller.

Appelman, Michiel, and Maikel de Boer. **"Performance analysis of OpenFlow hardware."** *University of Amsterdam, Tech. Rep* (2012).

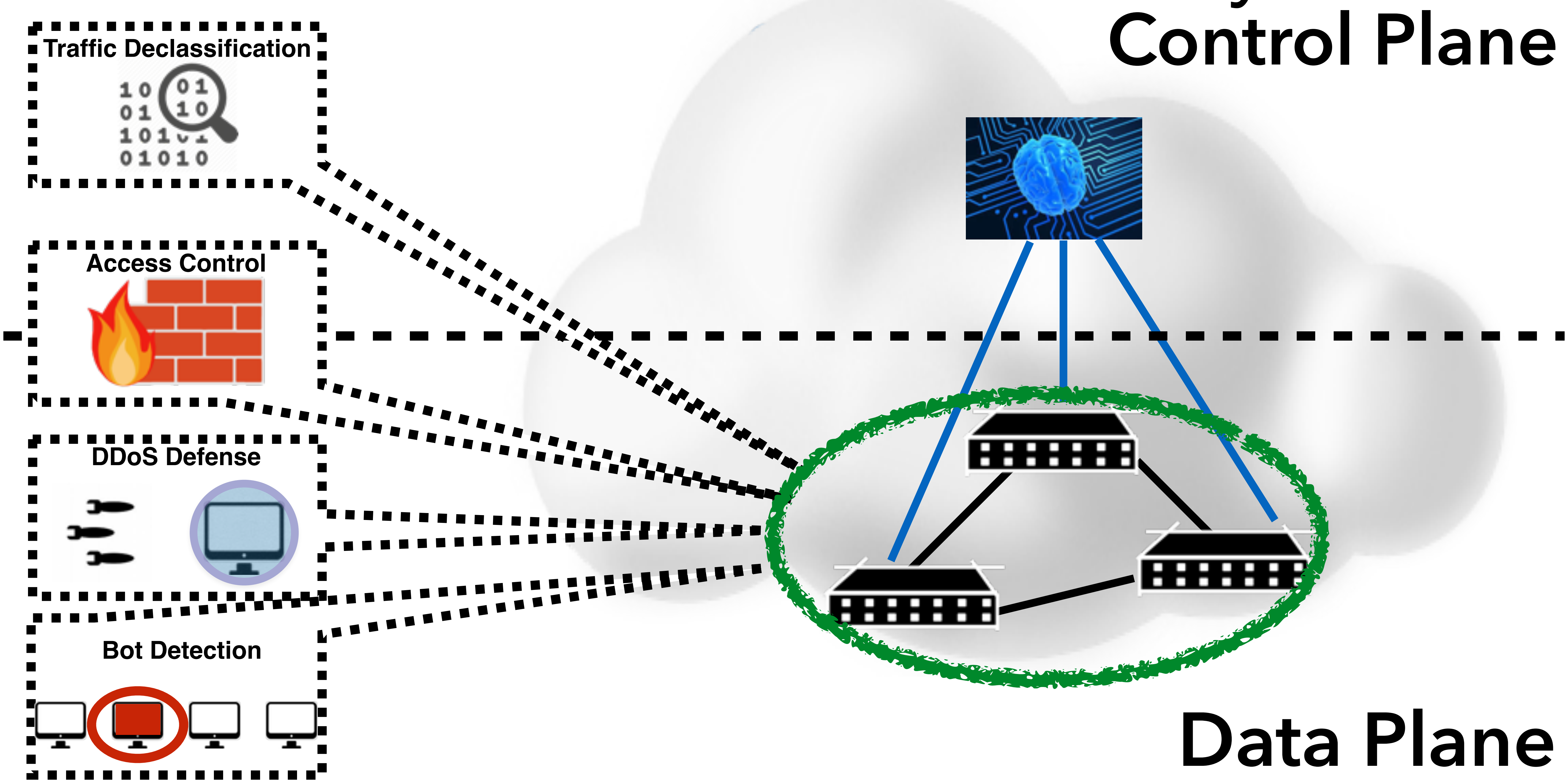
Curtis, Andrew R., et al. **"DevoFlow: scaling flow management for high-performance networks."** *ACM SIGCOMM Computer Communication Review*. Vol. 41. No. 4. ACM, 2011.

Obstacle: Centralized Control Plane

Our question: How Can We Make SDNs More Practical?



The General Approach: Switch Level Security



Control Plane

Traffic Declassification

Access Control

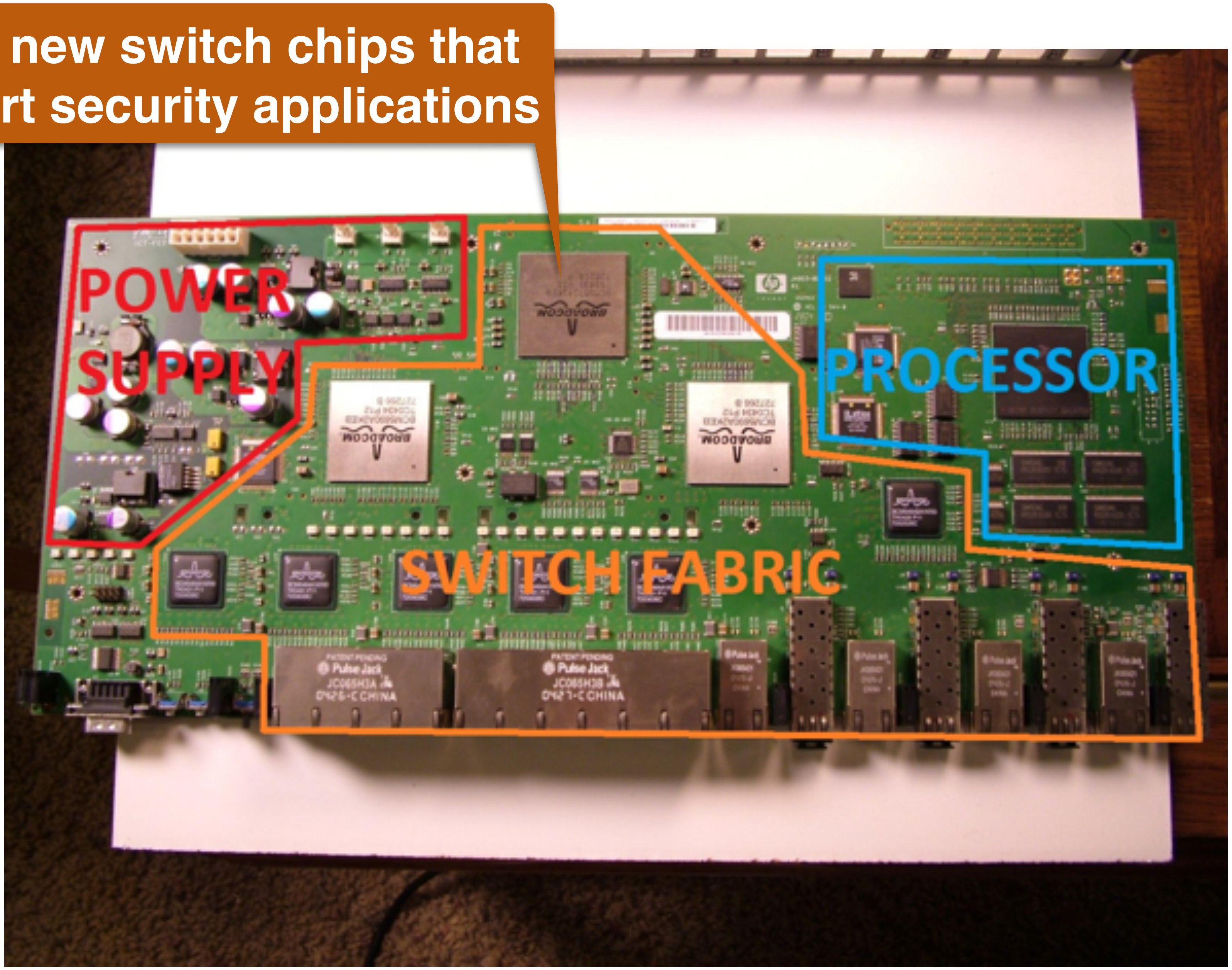
DDoS Defense

Bot Detection

Data Plane

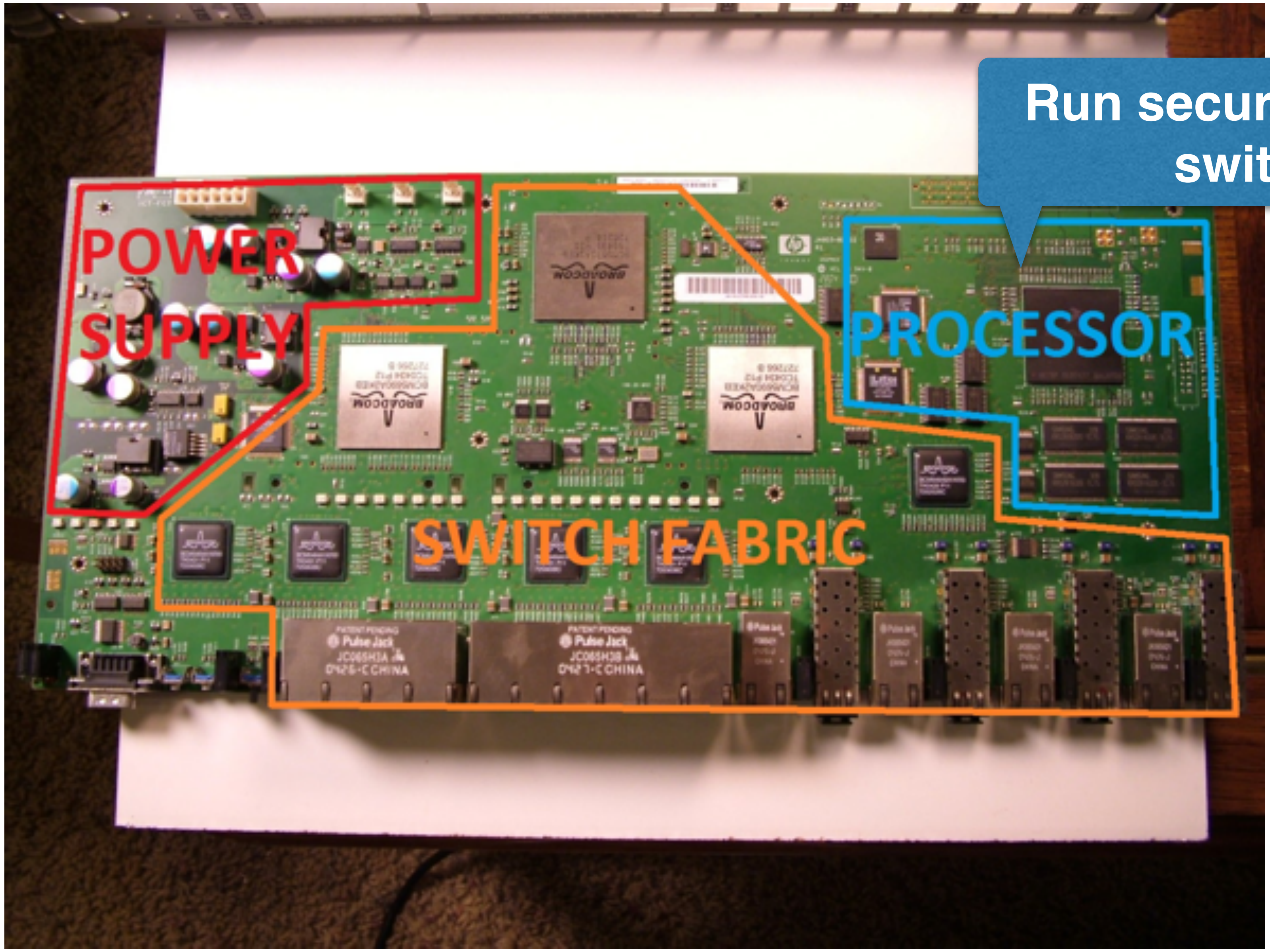
Previous Work: Security Functionality in the Forwarding Engine

Build new switch chips that support security applications



Shin, Seungwon, et al. **"Avant-guard: Scalable and vigilant switch flow management in software-defined networks."** *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013.

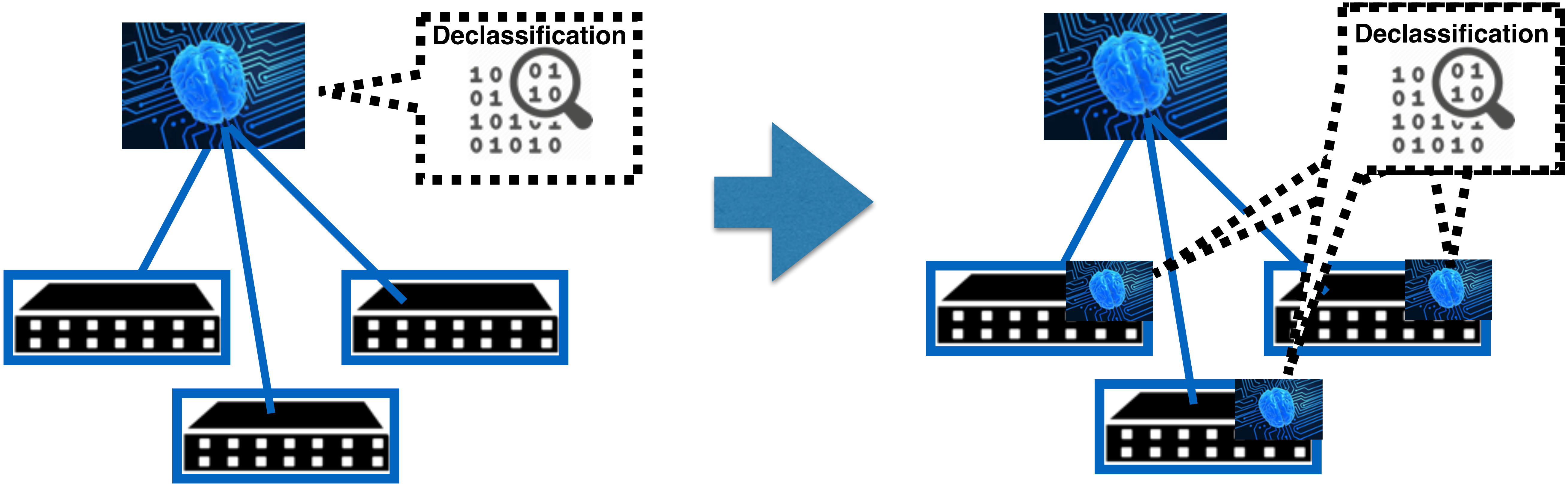
Our insight: Leverage Switch CPUs



Run security logic on the switch CPUs

OFX: A Framework for Application-Specific Switch Extensions

Each application can load custom functionality into switches. At runtime!



Outline

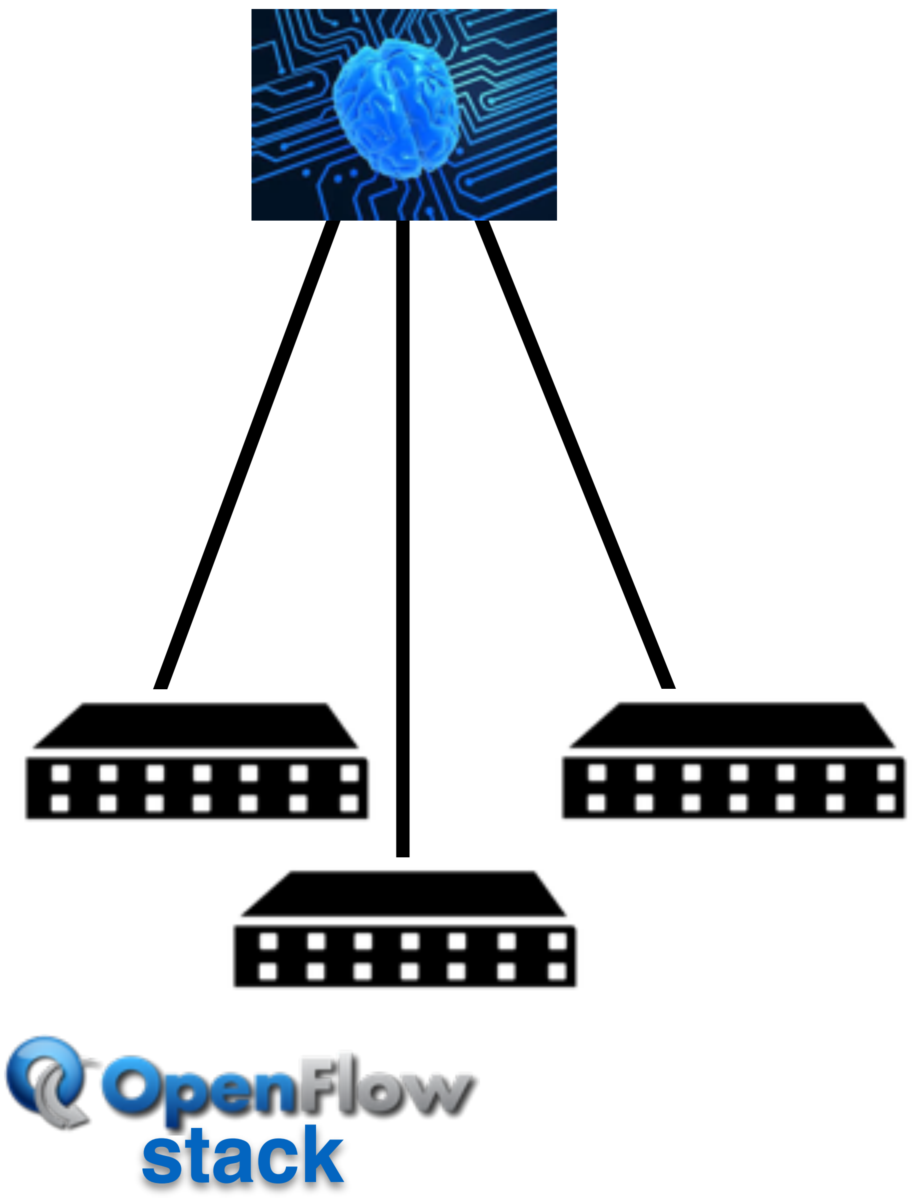
Introduction

Overview of OFX

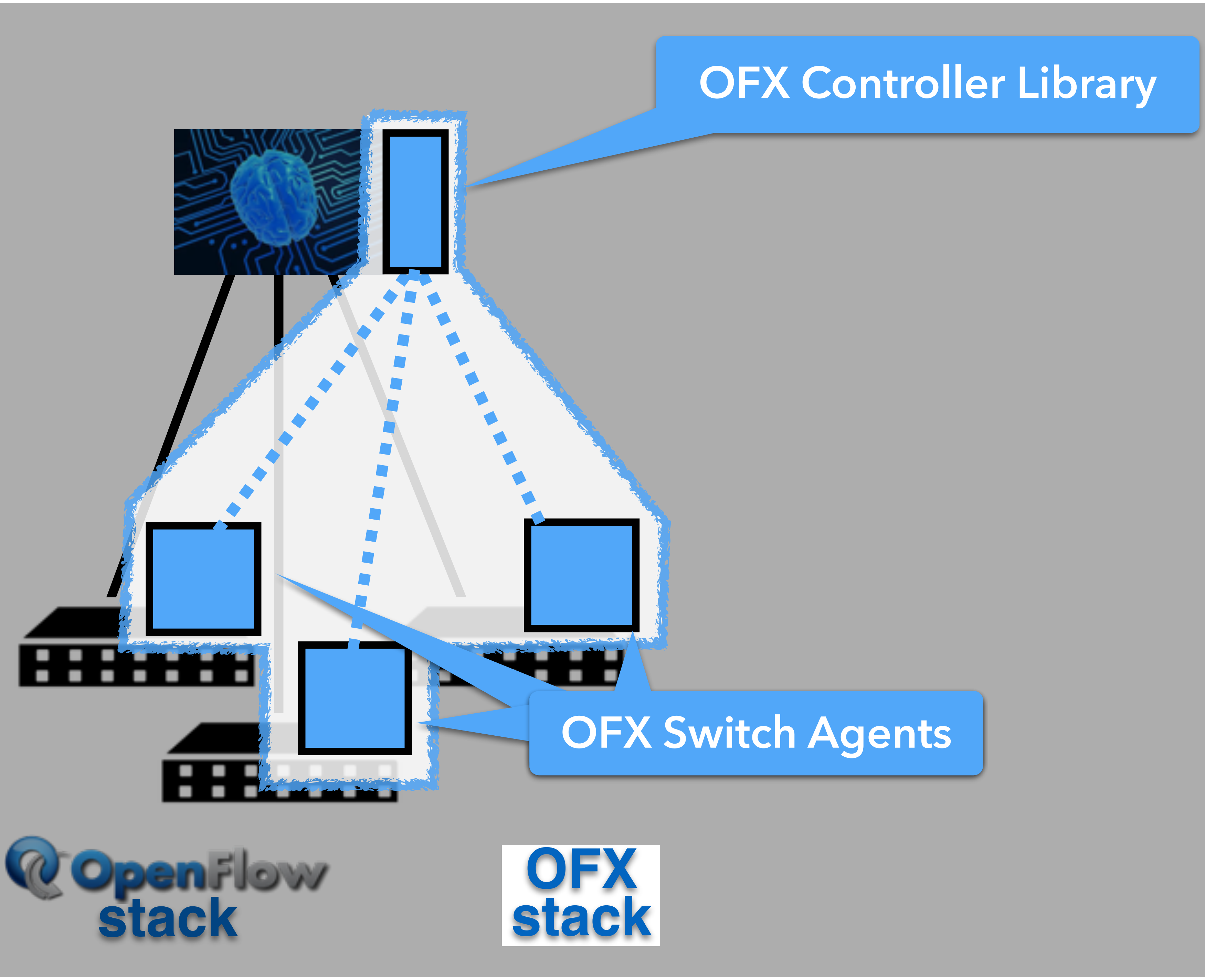
Using OFX

Benchmarks

OFX at a High Level



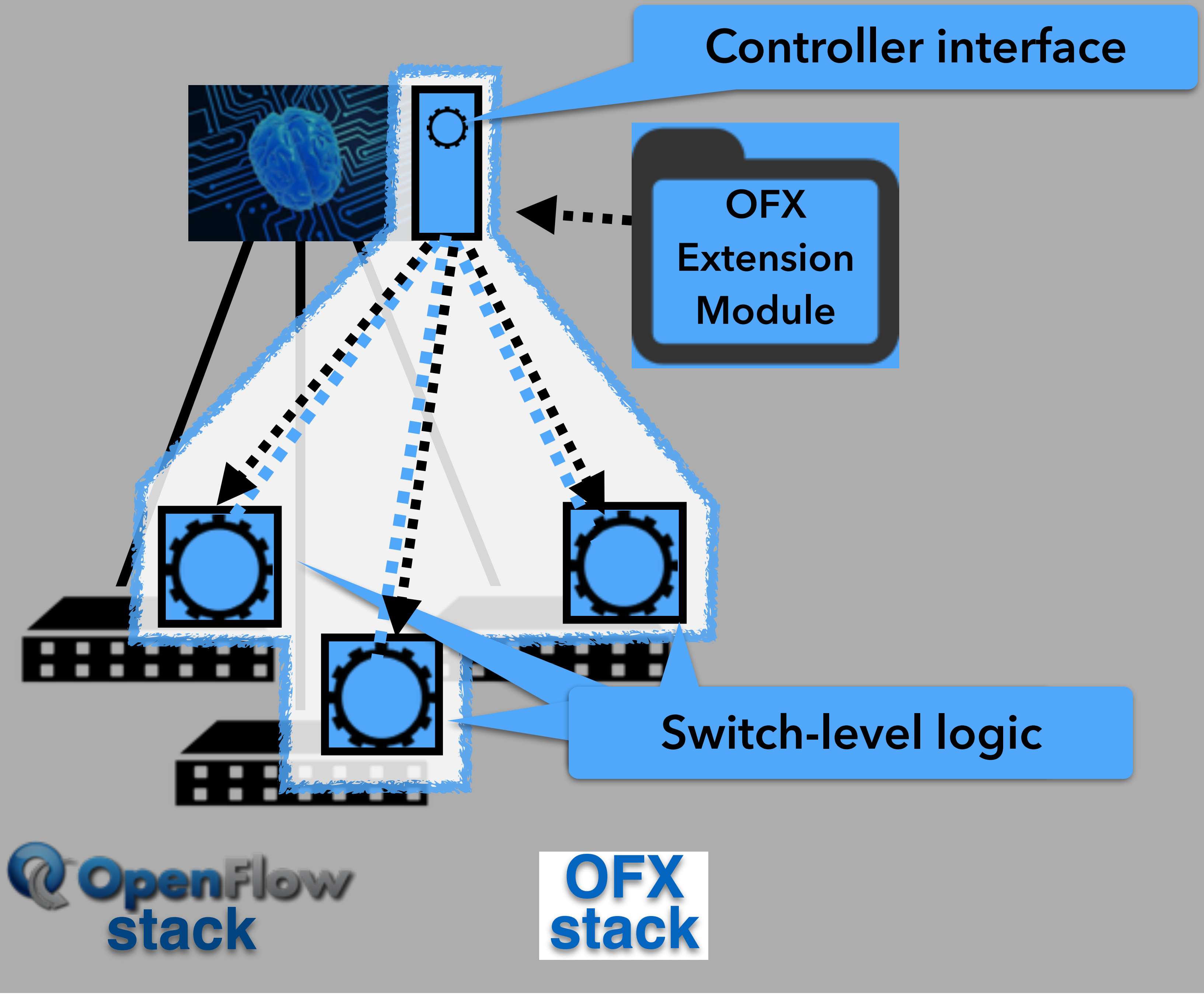
OFX at a High Level



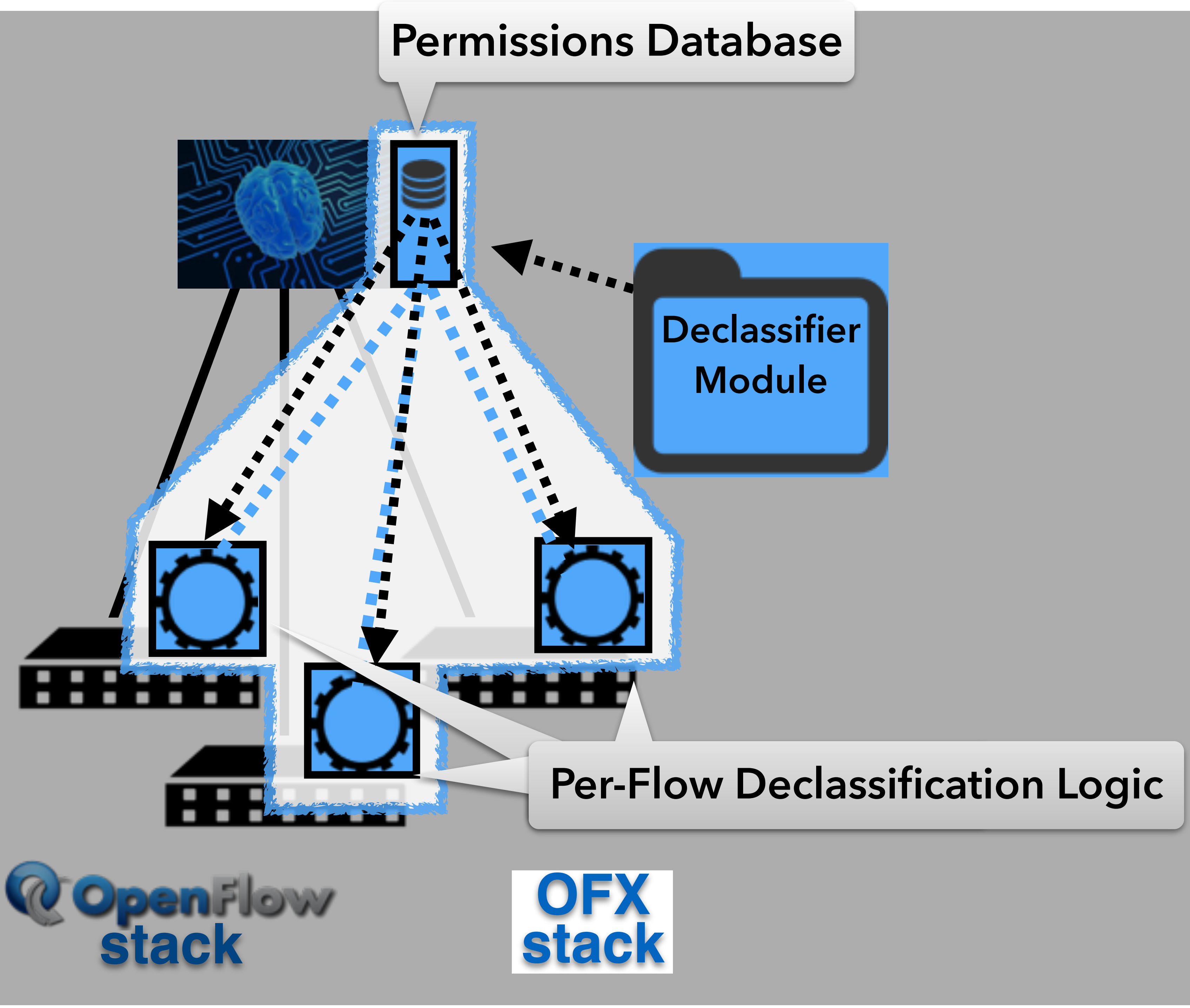
OFX Controller Library

OFX Switch Agents

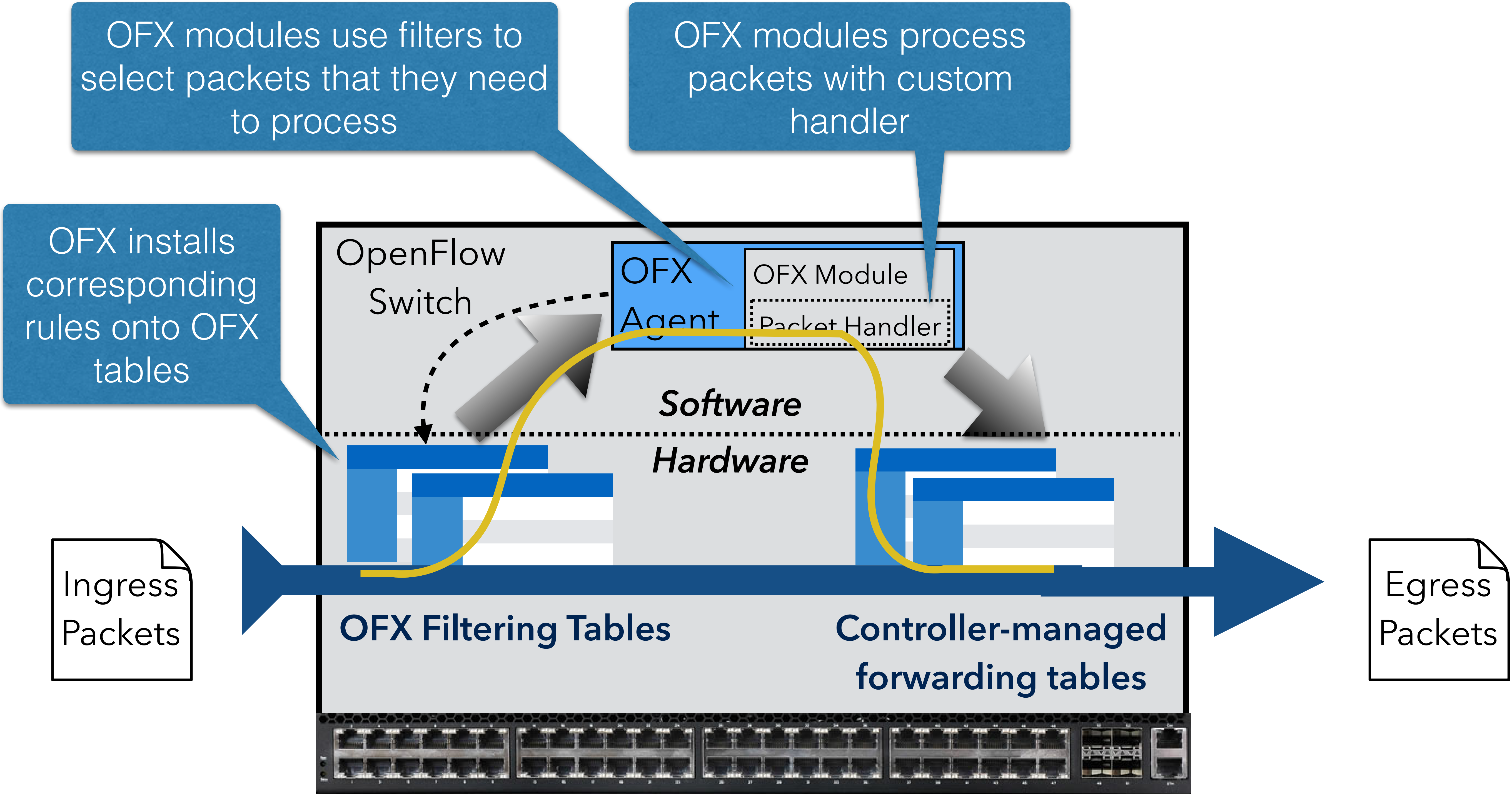
OFX at a High Level



OFX at a High Level



OFX at the Switch Level



OFX modules use filters to select packets that they need to process

OFX modules process packets with custom handler

OFX installs corresponding rules onto OFX tables

Outline

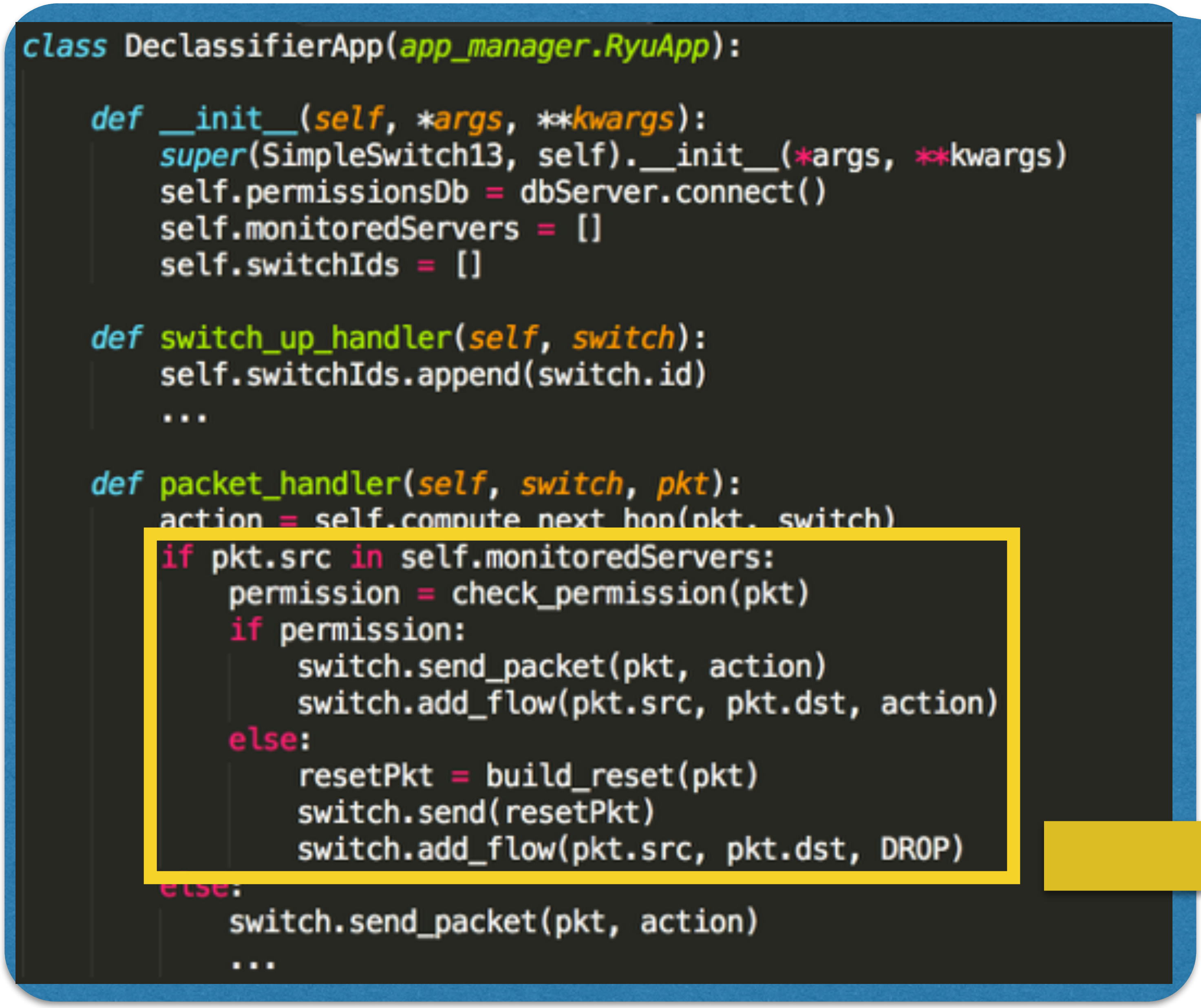
Introduction

Overview of OFX

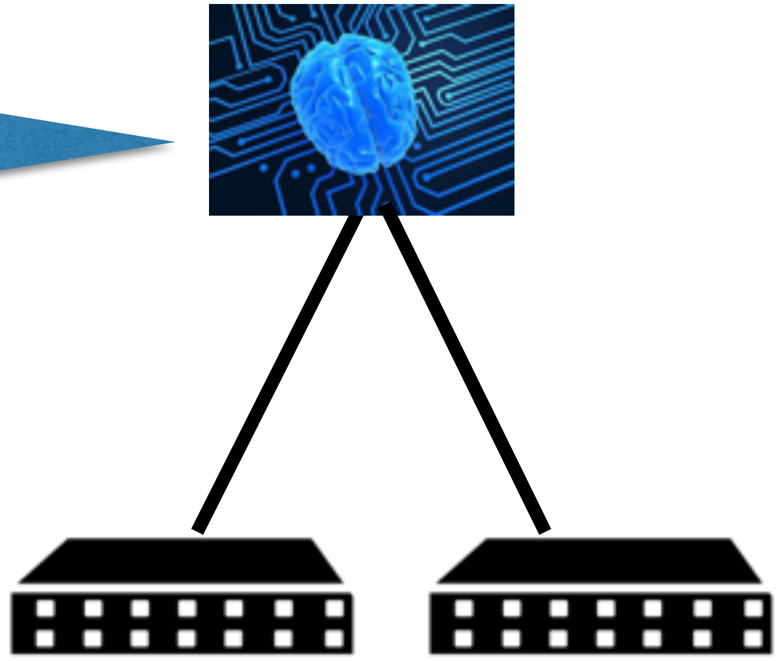
Using OFX

Benchmarks

Refactoring OpenFlow Applications to use OFX

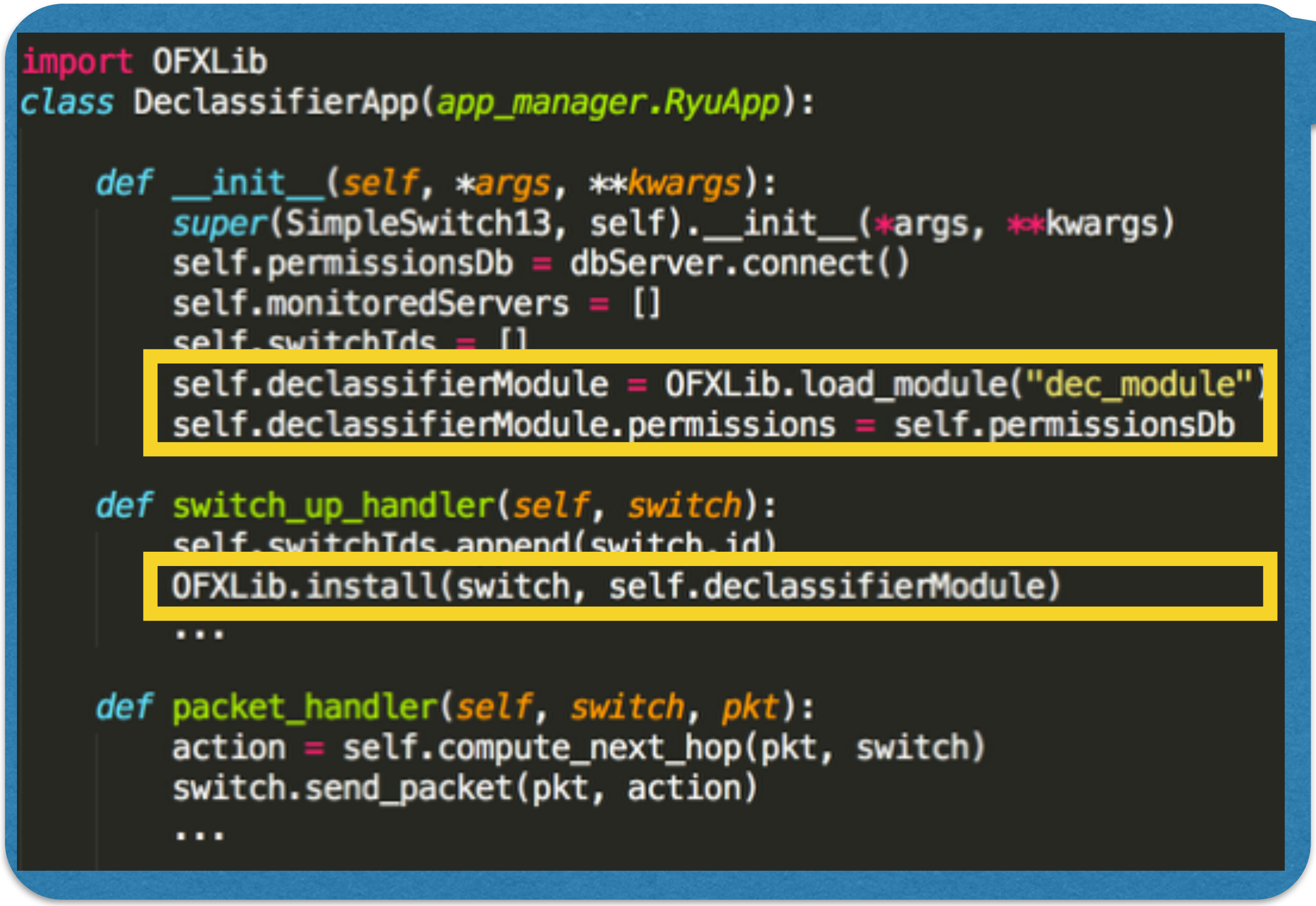


```
class DeclassifierApp(app_manager.RyuApp):

    def __init__(self, *args, **kwargs):
        super(SimpleSwitch13, self).__init__(*args, **kwargs)
        self.permissionsDb = dbServer.connect()
        self.monitoredServers = []
        self.switchIds = []

    def switch_up_handler(self, switch):
        self.switchIds.append(switch.id)
        ...

    def packet_handler(self, switch, pkt):
        action = self.compute_next_hop(pkt, switch)
        if pkt.src in self.monitoredServers:
            permission = check_permission(pkt)
            if permission:
                switch.send_packet(pkt, action)
                switch.add_flow(pkt.src, pkt.dst, action)
            else:
                resetPkt = build_reset(pkt)
                switch.send(resetPkt)
                switch.add_flow(pkt.src, pkt.dst, DROP)
        else:
            switch.send_packet(pkt, action)
            ...
```

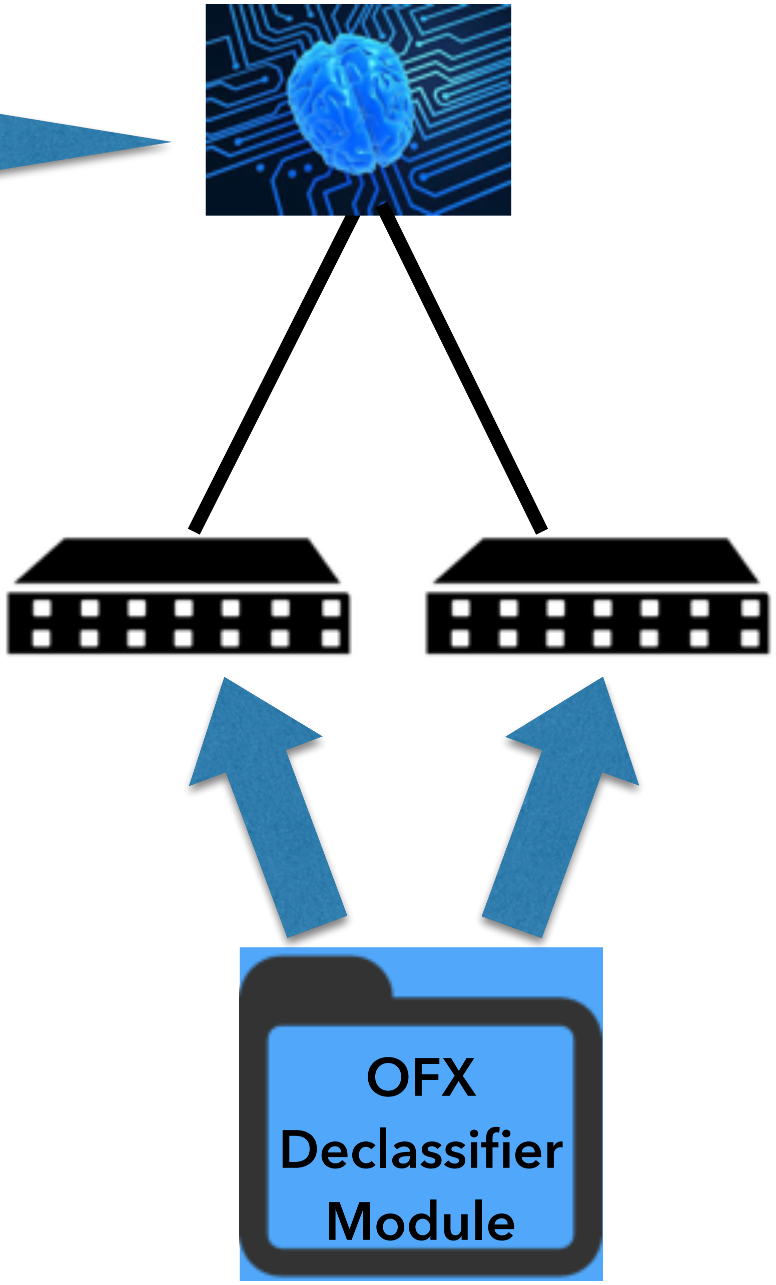


OFX Declassifier Module

Refactoring OpenFlow Applications to use OFX



```
import OFXLib
class DeclassifierApp(app_manager.RyuApp):

    def __init__(self, *args, **kwargs):
        super(SimpleSwitch13, self).__init__(*args, **kwargs)
        self.permissionsDb = dbServer.connect()
        self.monitoredServers = []
        self.switchIds = []
        self.declassifierModule = OFXLib.load_module("dec_module")
        self.declassifierModule.permissions = self.permissionsDb

    def switch_up_handler(self, switch):
        self.switchIds.append(switch.id)
        OFXLib.install(switch, self.declassifierModule)
        ...

    def packet_handler(self, switch, pkt):
        action = self.compute_next_hop(pkt, switch)
        switch.send_packet(pkt, action)
        ...
```



Outline

Introduction

Overview of OFX

Using OFX

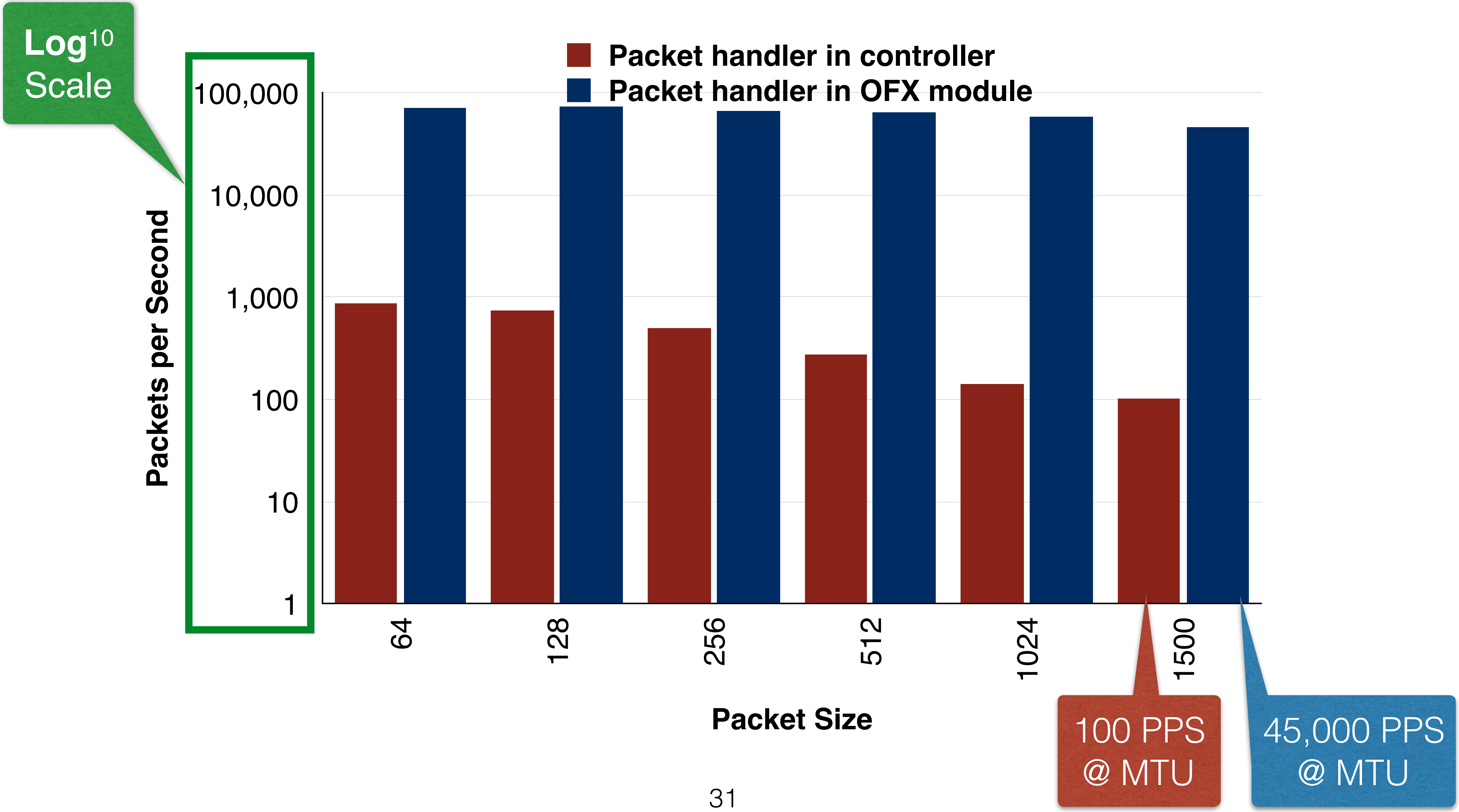
Benchmarks

Benchmarking OFX

How much raw overhead is there for processing packets with OFX?

How do OFX based security applications perform, compared with Middlebox and OpenFlow implementations?

OFX Benchmark: Packets Per Second



Benchmarking OFX

How much raw overhead is there for processing packets with OFX?

How do OFX based security applications perform, compared with Middlebox and OpenFlow implementations?

Benchmark: Declassifier Packet Drop Rate

Implementation	Frequent arriving flows	Median	High bandwidth flows
Middlebox Proxy	0.1%	0.1%	20.4%
OpenFlow	97.5%	88.2%	0.1%
OFX	5.1%	3.2%	0.1%

OpenFlow implementation limited by flow arrival rate

Proxy implementation limited by bit rate

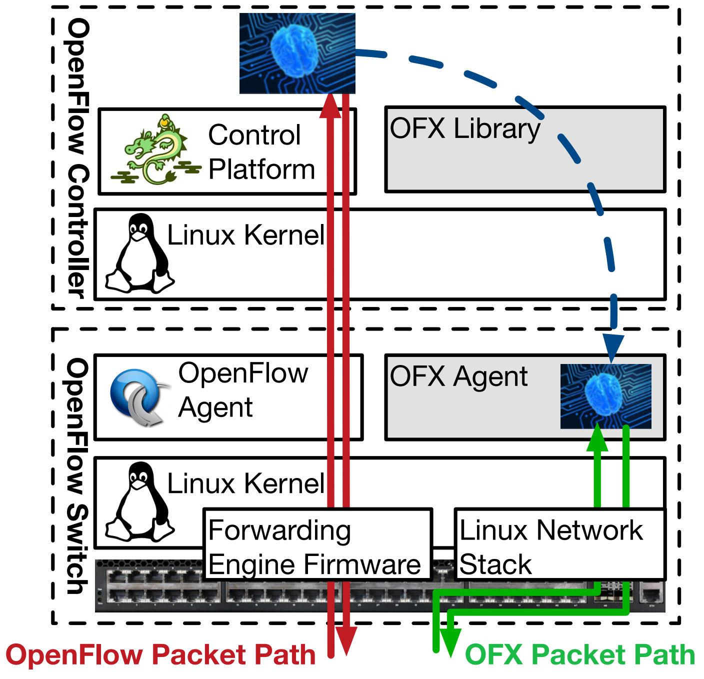
OFX implementation performed well in all workloads

Workload Name	Frequently arriving flows	Median flows	High bandwidth flows
Flow Inter-arrival Period	0.0015 Seconds	0.015 Seconds	0.15 Seconds
Average Transmission Bandwidth	19.75 Mbps	43.57 Mbps	970.99 Mbps

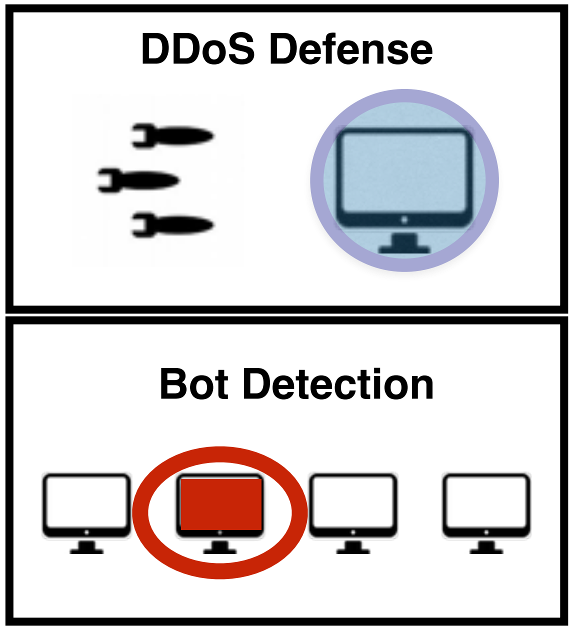
- S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, “**The nature of data center traffic: measurements & analysis,**” in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 2009, pp. 202–208.
- L. Qian and B. E. Carpenter, **“A flow-based performance analysis of tcp and tcp applications,”** in *Networks (ICON), 2012 18th IEEE International Conference on*. IEEE, 2012, pp. 41–45.

In the Paper

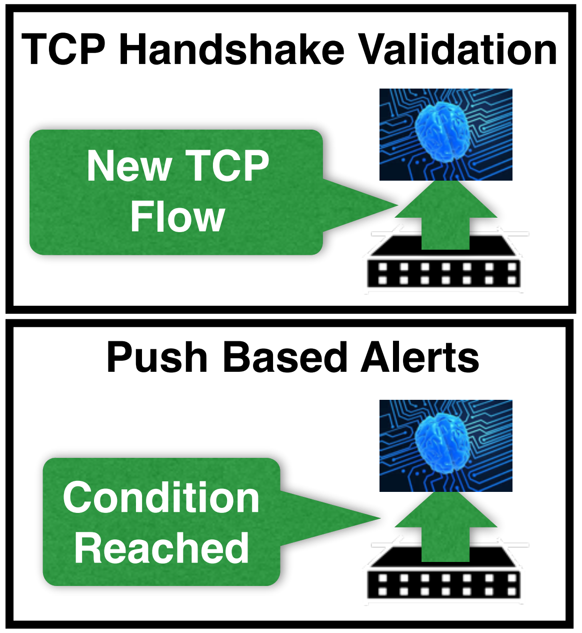
OFX API and Implementation Details



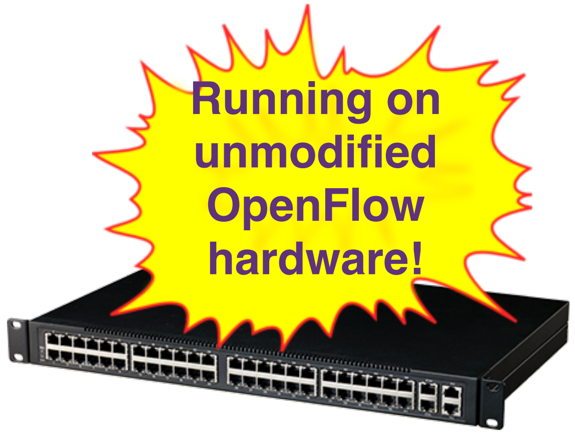
Application Specific Modules



Enhanced Switch API Modules

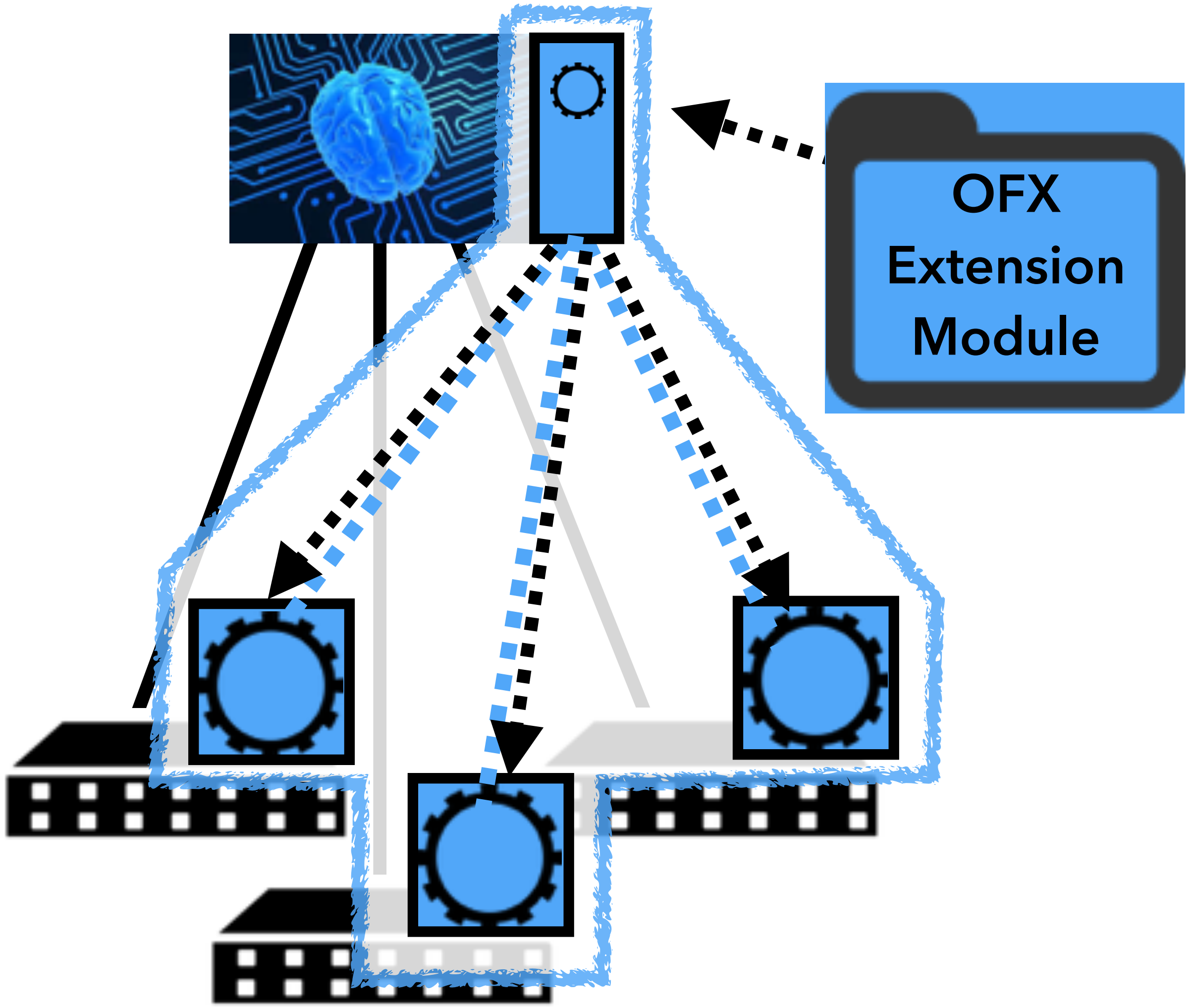


More benchmarks



Thank You

OFX: The OpenFlow Extension Framework



OFX lets OpenFlow security applications **push parts of their control plane logic down to switch CPUs**, which can greatly **improve performance and scalability on existing hardware and software.**

Penn
UNIVERSITY of PENNSYLVANIA

University of Colorado
Boulder